Methods and tools for automated adaptive traffic light control based on computer vision

Vasyl Teslyuk [1,†], Iryna Gado [1,*,†], Bohdan Fylypchuk [1,2,†] and Andrii Koval [1,†]

[1] Department of Automated Control Systems, Lviv Polytechnic National University, 12 S. Bandera Str., Lviv, Ukraine
[2] SoftServe, 2D Sadova Street, Lviv, 79021, Ukraine

Abstract

The paper presents a method for automated adaptive traffic light control in urban infrastructure using computer vision technologies. The proposed approach addresses pressing issues of traffic congestion, delays, and limited accessibility resulting from increasing vehicle density in modern cities. A modular and scalable software system has been developed to detect vehicles in real time using the YOLOv11 deep learning model, process the data through decision-making logic implemented in .NET Core. WebSocket is used for real-time communication between modules, while an automatic fallback to HTTP ensures continuity in case of connection loss. A React-based web interface allows for system monitoring, configuration management, and access to logs. A formal mathematical model is introduced to dynamically allocate green light durations based on real-time vehicle detection and configurable traffic density thresholds. Unlike traditional fixed-cycle systems or computationally heavy machine learning frameworks, the proposed solution balances precision, modularity, and responsiveness. The approach also anticipates future enhancements, including pedestrian detection for inclusive mobility and integration with smart city platforms.

Keywords

adaptive traffic light control, computer vision, YOLO, WebSocket, .NET Core, React, intelligent transportation systems

1. Introduction

The rapid growth in the number of vehicles in urban areas leads to the overloading of road infrastructure, increased traffic congestion, higher levels of air pollution, and delays in the movement of public transport and emergency services. These challenges drive the development and implementation of intelligent traffic management systems, particularly adaptive traffic lights capable of responding to real-time traffic conditions.

Modern approaches to dynamic traffic regulation increasingly rely on technologies such as computer vision, deep learning, and real-time video stream processing. However, the deployment of such systems often requires significant computational resources (especially GPUs), large volumes of training data, and complex infrastructure, which can hinder scalability in environments with limited budgets and technical capabilities.

This paper proposes a modular architecture for an adaptive traffic light system integrating a YOLO-based vehicle detection module, a WebSocket communication channel, decision-making logic implemented on the .NET platform, and an administrative panel developed with React. The proposed approach combines the high object detection accuracy typical of deep convolutional neural networks with the flexibility of modern web development technologies, thereby minimizing resource requirements and facilitating system customization, extension, and scalability.

The designed system operates in real time, dynamically adjusting traffic light phases based on current traffic density. To address broader urban mobility challenges, future developments are

*MoDaST 2025: Modern Data Science Technologies Doctoral Consortium, June, 15, 2025, Lviv, Ukraine

1,* Corresponding author.
† These authors contributed equally.
vasyl.m.teslyuk@lpnu.ua (V. Teslyuk); iryna.v.nychai@lpnu.ua (I. Gado); bohdan.fylypchuk.kn.2021@lpnu.ua (B. Fylypchuk); andrii.v.koval@lpnu.ua (A. Koval)
0000-0002-5974-9310 (V. Teslyuk); 0000-0003-1615-6483 (I. Gado); 0009-0006-5565-162X (B. Fylypchuk); 0009-0006-8815-1031 (A. Koval)

planned to incorporate pedestrian detection near crosswalks, enabling adaptive adjustments of traffic light phases. With the expansion of observation zones and further refinement of decision-making algorithms, the system could be adapted to promote inclusive urban environments, particularly to assist individuals with mobility impairments — a demographic that has significantly increased as a result of the full-scale war in Ukraine. The integration of automated pedestrian detection would enhance safety and contribute to reducing physical barriers within urban infrastructure.

The proposed study aims to automate the process of adaptive traffic light control within urban transport environments by employing computer vision technologies, enabling real-time vehicle detection. The developed system is designed to enhance traffic management efficiency, reduce congestion, and improve the throughput of intersections under dynamic regulation conditions.

The object of the study is the traffic light regulation process within urban infrastructure.

The subject of the study is the methods and tools for automated adaptive traffic light control based on computer vision technologies and software-hardware solutions.

The main objective of the work is to increase the efficiency of traffic flow management by developing a model for automated traffic light regulation using computer vision technologies and modern real-time data processing tools.

2. Related Works

In contemporary scientific and engineering literature, intelligent traffic control systems utilizing computer vision and deep learning are actively studied. In particular, the works [1–3] propose models for adaptive traffic light control based on the prediction of traffic flows using neural networks.

Deep reinforcement learning algorithms for optimizing traffic light cycles under varying traffic intensities are discussed in [4–6]. In contrast, studies [7, 8] focus on the application of computer vision for vehicle detection, which serves as the basis for decision-making regarding signal phase changes. However, such systems often require significant computational resources and involve complex deployment procedures.

Solutions based on OpenCV and the processing of regions of interest (ROIs) are described in [9, 10], where the authors emphasize the efficiency of real-time vehicle detection under constrained computational resources.

A distinct category of research [11, 12] addresses the theoretical aspects of adaptive traffic light control, as well as hybrid approaches that combine classical algorithms with machine learning methods.

Against this backdrop, the proposed study is particularly relevant as it focuses not only on the theoretical analysis of existing approaches but also on the practical integration of key components into a unified system. Specifically, the integration of YOLO as a vehicle detector, a WebSocket channel for real-time communication, control logic implemented on the .NET platform, and an administrative interface developed with React enables the creation of an adaptive system that operates with minimal latency, is scalable, and does not require complex retraining procedures.

Compared to systems based solely on complex machine learning models, the proposed solution demonstrates a balance between accuracy, flexibility, and computational efficiency, making it suitable for rapid pilot deployment in urban environments.

3. Materials and Methods

Inefficient traffic management based on fixed traffic light cycles often leads to traffic congestion, excessive fuel consumption, and increased emissions. The inability to adapt to fluctuations in traffic volume decreases the overall efficiency of the transport system. The solution to this issue is the implementation of intelligent traffic management systems, combining computer vision data with adaptive decision-making algorithms.

The development of the intelligent dynamic traffic light management system required a comprehensive technical approach, which included a well-founded choice of system architecture, technologies, and implementation tools. The primary criteria for the selection of these components were stable real-time operation, scalability, deployability on available hardware, and ease of future maintenance.

A modular system architecture was designed to incorporate components for computer vision, traffic light control logic, communication channels between system modules, an administrative interface, and a data storage system. Python was selected as the primary programming language for implementing the computer vision module due to its flexibility and compatibility with libraries such as OpenCV, NumPy, and Torch, as well as with modern image processing models, including Ultralytics YOLO. This made real-time video processing feasible even on embedded devices.

To support the object detection functionality, the YOLOv11 family of models developed by Ultralytics was chosen due to its balance between inference speed and detection accuracy [13]. These models were pretrained on the COCO dataset (Common Objects in Context), a widely adopted benchmark in computer vision research. A detailed summary of the YOLOv11 model variants is provided in Table 1, while the key characteristics of the COCO dataset are presented in Table 2.

Table 1
Comparison of YOLO11 Models on the COCO Dataset [13]

Model	Image Size	mAP (50–95)	CPU ONNX (ms)	T4 TensorRT (ms)	Params (M)	FLOPs (B)
YOLO11n	640	39.5	56.1 ± 0.8	1.5 ± 0.0	2.6	6.5
YOLO11s	640	47.0	90.0 ± 1.2	2.5 ± 0.0	9.4	21.5
YOLO11m	640	51.5	183.2 ± 2.0	4.7 ± 0.1	20.1	68.0
YOLO11l	640	53.4	238.6 ± 1.4	6.2 ± 0.1	25.3	86.9
YOLO11x	640	54.7	462.8 ± 6.7	11.3 ± 0.2	56.9	194.9

Table 2
COCO Dataset Characteristics [14]

Parameter	Value
Total Images	330,000+
Annotated Images	200,000+
Object Categories	80
Annotations	1.5 million+
Types of Tasks	Object Detection, Segmentation, Captioning
Common Classes	Person, Car, Dog, Chair, etc.
Format	JSON
License	Creative Commons Attribution 4.0

A performance comparison of YOLOv11 models in terms of accuracy and inference time is illustrated in **Figure 1**.

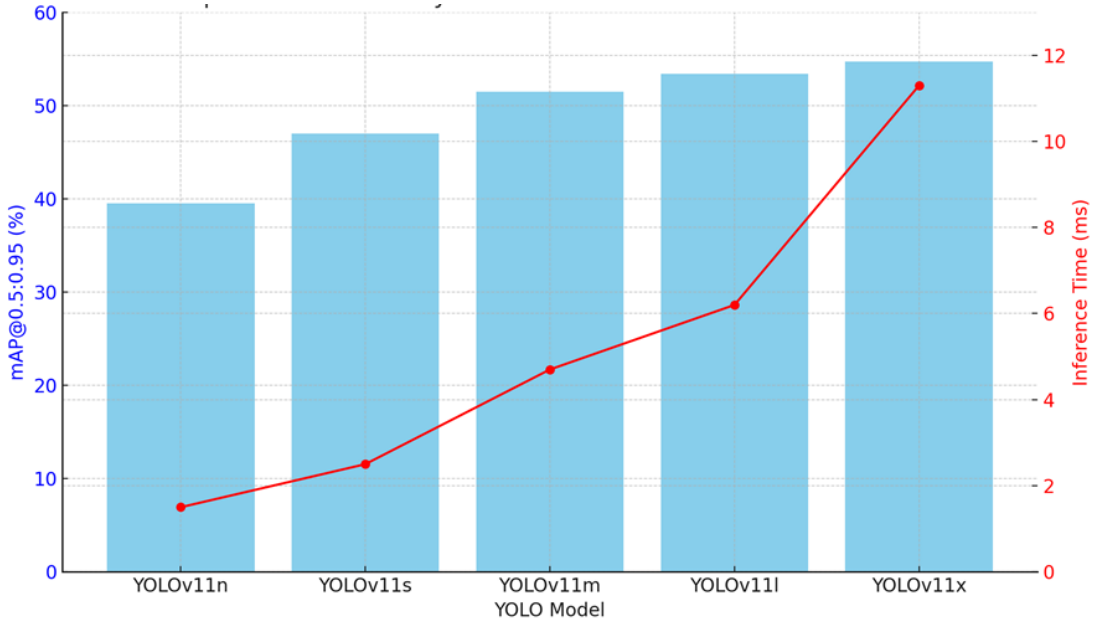

Figure 1: Comparison of YOLOv11 model variants in terms of mean Average Precision (mAP) and inference time on the COCO dataset.

The server-side component of the system was developed using the .NET Core platform and C#, which allows for efficient event processing, adaptive traffic light control, interaction with the MSSQL database, and API provision for the client-side. The administrative interface was built using React, allowing system monitoring, configuration parameter adjustments, log viewing, and intersection status updates.

For real-time data exchange between the Python application and the server-side, WebSocket was employed, ensuring stable two-way communication. If the connection is lost, the system automatically switches to a backup HTTP channel to ensure continuous operation. The MSSQL relational database is used to store traffic light configurations, change histories, and action logs. Docker was utilized for containerization of system components, while Git was used for version control and collaborative development.

The selected technologies complement each other effectively, ensuring that all functional system requirements are met—from video data collection and event processing to adaptive traffic light control and administration. This approach ensures stable system operation even with limited hardware resources and provides a foundation for future scalability and integration with other city infrastructure systems.

To ensure real-time adaptive control of traffic lights, the system implements a calculation model that dynamically determines the duration of the green phase based on the detected number of vehicles. The underlying algorithm is described below.

The system determines the duration of the green phase for each traffic light based on a direct dependency model linked to the number of detected vehicles.

Using real-time computer vision data, the number of vehicles N approaching from each direction is calculated. To improve detection reliability, a confidence threshold $c_{threshold}$ is introduced, which defines the minimum confidence level required for an object to be considered a valid detection.

An object is recognized as a vehicle and counted only if the following condition is met:

$$c_{object} \geq c_{threshold}, \quad (1)$$

where c_{object} is the model's confidence in the detection, $c_{threshold}$ is the predefined confidence threshold (set to 0.7 in the current implementation).

Thus, the effective number of detected vehicles N is calculated as:

$$N = \sum_{i=1}^{M} \delta(c_i \geq c_{threshold}), \quad (2)$$

where M – the total number of detected objects in the frame, c_i – the confidence value for the i-th detection, $\delta(\cdot)$ – the indicator function, equal to 1 if the condition is true and 0 otherwise.

The resulting number N is then utilized within the proposed model to calculate the adaptive green phase duration, ensuring consistency with the system's real-time decision-making framework.

A sequence table (SequenceGreenTime) is configured by the system administrator, specifying recommended green phase durations T for different ranges of vehicle counts N. This approach enables the consideration of specific characteristics of individual intersections and local traffic patterns (for instance, by allocating additional time for accident-prone directions or transit routes).

If an exact match is found in the *SequenceGreenTime(N)* table, the predefined duration is used:

$$T = SeqeunceGreenTime(N), \quad (3)$$

If no exact match exists, the green phase duration is calculated according to the following formula:

$$T = N \times t_{per\_vehicle}, \quad (4)$$

where $t_{per\_vehicle}$ represents the standard time allocated per vehicle.

The calculated value is then constrained within administrator-defined minimum and maximum bounds:

$$T_{min} \leq T \leq T_{max}, \quad (5)$$

where T_{min} and T_{max} are administrator-defined parameters set in the configuration settings.

The final green phase duration is thus determined as:

$$T = max\left(T_{min}, min\left(T_{max}, SequenceGreenTime(N) \vee N \times t_{per\_vehicle}\right)\right) \quad (6)$$

The resulting value T is transmitted via WebSocket to the controlled group of traffic lights, updating their operating mode in real time.

4. Results

For the purpose of demonstrating the architecture of the developed system, two separate class diagrams were created: one for the server-side module built on the .NET Core platform, and another for the client-side application developed in Python, responsible for implementing the computer vision functionality. Figure 2 presents the Python-based diagram, as it provides a more concise and illustrative representation of the client-side application’s core functionality.

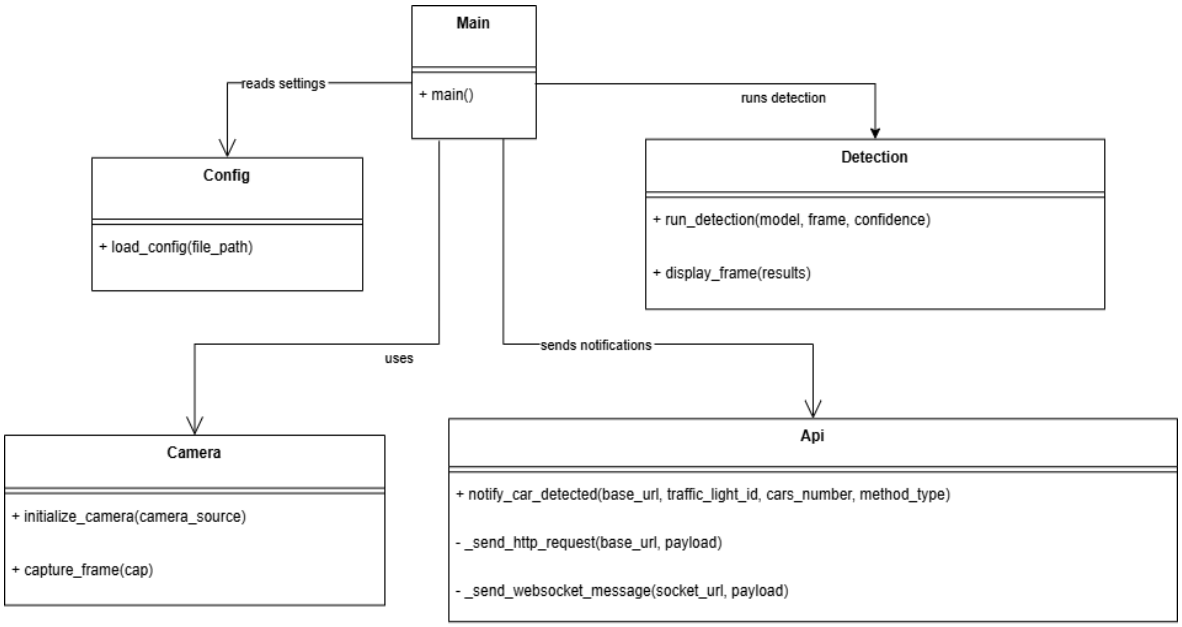

Figure 2: Class diagram of the client-side Python application for vehicle detection in the computer vision system.

At the core of the system's functional logic lies the vehicle detection use case, which initiates the adaptive control of traffic light phases. This process defines the starting point for the real-time traffic analysis cycle.

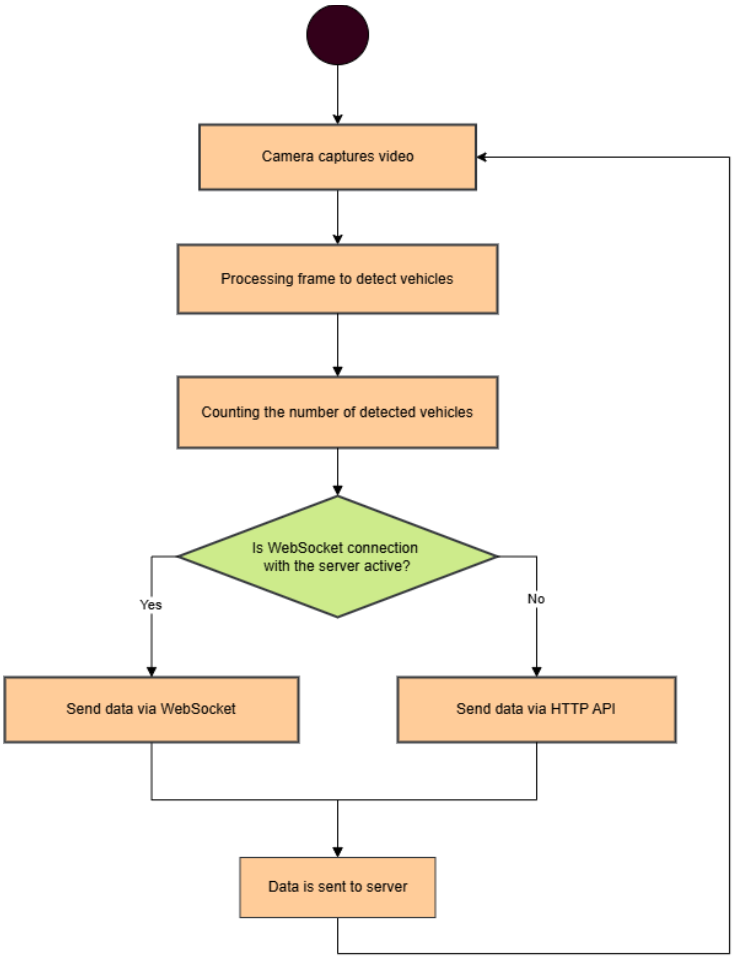

Figure 3: Activity diagram for the "Vehicle Detection" use case.

The vehicle detection process is implemented at the client-side application level, where the incoming video stream from the camera is processed using the YOLO deep neural network. This network enables accurate and rapid identification of vehicles in video frames. The detection results are aggregated into a message containing the number of detected vehicles, the camera identifier, the timestamp, and other relevant parameters. This message is transmitted to the server via a WebSocket connection and is used for decision-making regarding traffic signal changes.

Thus, the vehicle detection event serves as the starting point for the operation of the entire intelligent system: it ensures the acquisition of primary data, forms the basis for analytical decisions, and determines the efficiency of response to changes in traffic load.

For a comprehensive representation of the developed system's architecture and the interaction among its modules, a component diagram was constructed. The diagram depicts the communication flows between the computer vision application, the backend server, the database, and the administrative panel. This integrated model emphasizes the modular structure of the system and its real-time data processing capabilities in the context of dynamic traffic light control.

Figure 4 illustrates the overall system architecture, emphasizing the modular components and their communication flows.

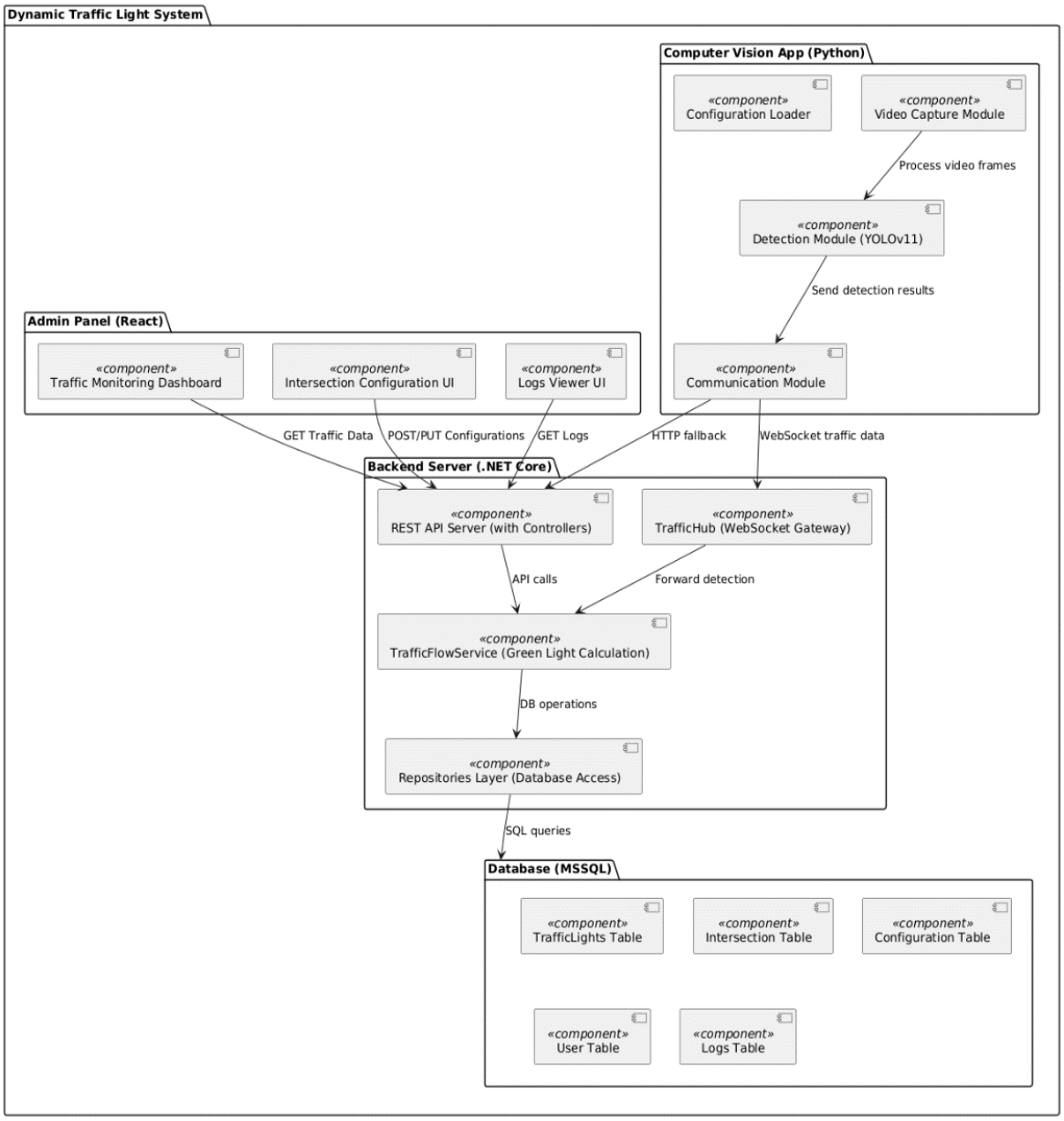

Figure 4: Component diagram of the dynamic traffic light system architecture.

The functioning of the system was tested in a simulation environment using video data from surveillance cameras. The results confirmed the consistency of the algorithm and server-side logic.

To assess the responsiveness of the traffic light control system, an experimental delay measurement was conducted for two types of communication. In the HTTP-based architecture, the vehicle detector sends a request to the server over HTTP, which then relays a command to the traffic light over a persistent socket connection. In the socket-based architecture, the detector communicates directly with the server via a socket, which then forwards the command through another socket to the traffic light. Measurements were taken locally, with minimal external network interference, in order to accurately capture the internal latency of each architecture.

The results demonstrate that the HTTP-based approach exhibited average delays typically ranging from 17 to 26 milliseconds after initial setup. These values are acceptable for a fallback communication channel in case the primary socket-based channel becomes unavailable. In contrast, the socket-based implementation consistently demonstrated lower and more stable latency, typically between 16 and 20 milliseconds, which is preferable for real-time system responsiveness. These patterns are clearly illustrated in Figure 5.

Figure 5: Latency measurements for two communication architectures used in traffic light control: HTTP-based and socket-based.

The table 3 summarizes the comparative characteristics of both communication models.

Table 3
Comparison of communication architectures in terms of latency

Architecture	Average Delay	Main Sources of Delay
HTTP	Higher, variable	Protocol overhead, possible new connections per request
Socket	Low, stable	Persistent connection, minimal protocol overhead

These findings confirm that while HTTP remains a viable backup option, the persistent socket connection substantially improves the system's reaction time and ensures higher efficiency under normal operating conditions.

5. Discussions

The adaptive traffic management system developed in this work is based on the application of computer vision technologies for real-time traffic situation assessment and dynamic adjustment of traffic light phases. This approach enables flexible and efficient control in contrast to classical methods.

Traditional systems like SCATS (Australia) and SCOOT (UK) use magnetic sensors and predefined algorithms, but lack flexibility and require costly infrastructure.

In parallel, modern open-source solutions are evolving, including projects such as SmartFlow, Byte-Blender, and Dacee-dee — all publicly available on GitHub. These projects focus on affordable vehicle detection from real-time video streams, mainly using Python and YOLO frameworks. However, they often exhibit limited server-side logic, simplified architectures, and lack deep adaptability for managing extensive traffic networks.

In contrast, our system combines YOLO-based vision in Python, .NET Core decision logic, and WebSocket communication to provide reliable, scalable control. Its hybrid architecture allows modularity and future extensions like forecasting or Smart City integration.

Despite its advantages, the chosen architectural approach has certain technical limitations. Since video processing and detection are handled by a separate Python application, the system depends on stable communication between Python and .NET components. In case of disconnection or errors, the .NET server may not receive real-time data for adaptive control.

WebSocket transmission requires robust handling of interruptions and reconnections, especially in unstable urban networks. Scaling to many cameras and lights may require server optimization to handle multiple connections.

Nonetheless, these disadvantages are outweighed by the system's enhanced stability, performance, and reliability—factors that are crucial for deployment in real-world urban environments. The system demonstrates a high level of engineering maturity, allows modular component isolation, and ensures the convenient integration of new functionalities via API or other standard interfaces.

This study advances the field of intelligent transportation systems by developing an adaptive traffic light control method that dynamically adjusts signal timings based on real-time computer vision analysis. A hybrid distributed system architecture is proposed, integrating a lightweight YOLO-based vehicle detection module implemented in Python with a robust server-side decision-making engine based on .NET Core technologies, ensuring efficient operation even with limited computational resources. The research formalizes a real-time prioritization model that calculates green phase durations dynamically, using confidence-filtered vehicle detections and administrator-configurable adaptation ranges. Furthermore, a resilient communication mechanism was implemented using WebSocket protocols with automatic fallback to HTTP, which guarantees uninterrupted data transmission under unstable network conditions typical of urban infrastructures. The modular and scalable system design also provides flexibility for future extensions, such as pedestrian detection integration or Smart City platform interoperability. By combining advanced AI-based detection techniques with distributed and fault-tolerant control logic, this work contributes a practical and scientifically grounded solution for the deployment of adaptive traffic light systems in real-world urban environments.

Thus, the proposed system successfully combines the theoretical foundations of adaptive traffic management with efficient engineering implementation, ensuring its competitiveness among existing solutions.

Conclusions

This paper proposes a scientifically grounded method for the automated adaptive control of traffic lights, based on computer vision technologies and modern real-time data processing tools. The developed system combines theoretical approaches to modeling traffic processes with practical engineering solutions focused on operational stability, scalability, and integration into urban infrastructure.

The modular architecture of the system enables efficient processing of video streams for vehicle detection, dynamic adjustment of green light durations based on current traffic density, and continuous communication between client-side and server-side modules via the WebSocket protocol.

In future development stages, the system may be enhanced with functionality for pedestrian recognition near crossings to improve inclusivity. This would allow the algorithm to be adapted to the needs of people with mobility impairments, for whom timely signal changes are crucial.

The results obtained lay the groundwork for deploying the system in real urban environments and further advancing intelligent transportation systems, considering Smart City principles and inclusive design. The developed solution demonstrates high potential for implementation under resource-constrained conditions, ensuring a balance between detection accuracy, processing speed, and technical reliability.

Thus, the research makes a significant contribution to the development of intelligent transportation systems, laying the foundation for further scientific research and practical implementation of adaptive urban traffic management technologies.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT for grammar and spelling checks, as well as for improving the clarity of certain passages. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

1. M. Ayoubi, H. Aman, R. Akbari, H. Lodin, "AI-enabled traffic light control system: An efficient model to manage the traffic at intersections using computer vision," *International Journal of Integrated Science and Technology*, vol. 2, no. 8, pp. 767–794, 2024, doi: 10.59890/ijist.v2i8.2438.
2. Y. Yang, "Deep learning-based detection for traffic control," in *Proceedings of the 5th International Conference on Advances in Artificial Intelligence (ICAAI 2021)*, ACM, 2021, pp. 1–9, doi: 10.1145/3505711.3505736.
3. K. S. Mehta, K. N. Raj, K. N. Brahmbhatt, "Machine learning solutions for adaptive traffic signal control: A review of image-based approaches," *World Journal of Advanced Engineering Technology and Sciences*, vol. 13, no. 1, pp. 476–481, 2024, doi: 10.30574/wjaets.2024.13.1.0437.
4. S. Agrawal, R. Sharma, P. Srivastava, V. Patel, "Adaptive Traffic Signal Control System Using Deep Reinforcement Learning," in *Proceedings of the 2024 IEEE International Conference on Intelligent Signal Processing and Effective Communication Technologies (INSPECT)*, Gwalior, India, 2024, pp. 1–6, doi: 10.1109/INSPECT63485.2024.10896157.
5. J. Luo, X. Li, Y. Zheng, "Researches on Intelligent Traffic Signal Control Based on Deep Reinforcement Learning," in *Proceedings of the 16th International Conference on Mobility, Sensing and Networking (MSN 2020)*, Tokyo, Japan, 2020, pp. 729–734, doi: 10.1109/MSN50589.2020.00124.
6. T. Mao, A.-S. Mihăită, F. Chen, H. L. Vu, "Boosted Genetic Algorithm Using Machine Learning for Traffic Control Optimization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7112–7141, 2022, doi: 10.1109/TITS.2021.3066958.

7. S. A. Anwar, F. T. Zohura, J. Paul, "Intelligent traffic control system using computer vision algorithms," in *Proceedings of SPIE 12673, Optics and Photonics for Information Processing XVII*, 2023, doi: 10.1117/12.2682676.
8. B. Chong, M. A. Ismail, "Smart traffic light control system using image processing," *IOP Conference Series: Materials Science and Engineering*, vol. 1088, no. 1, 012021, 2024, doi: 10.1088/1757-899X/1088/1/012021.
9. Z. Fahrunnisa, R. Rahmadwati, R. A. Setyawan, "Adaptive traffic light signal control using fuzzy logic based on real-time vehicle detection from video surveillance," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, vol. 10, no. 2, pp. 123–132, 2023, doi: 10.26555/jiteki.v10i2.28712.
10. A. P. Rangari, A. R. Chouthmol, C. Kadadas, P. Pal, S. K. Singh, "Deep Learning based smart traffic light system using Image Processing with YOLO v7," in *Proceedings of the 4th International Conference on Circuits, Control, Communication and Computing (I4C 2022)*, Bangalore, India, 2022, pp. 129–132, doi: 10.1109/I4C57141.2022.10057696.
11. T. Azfar, J. Li, H. Yu, R. L. Cheu, Y. Lv, R. Ke, "Deep learning-based computer vision methods for complex traffic environments perception: A review," *arXiv preprint*, arXiv:2211.05120, 2022.
12. S. Sun, H. Wu, L. Xiang, "City-Wide Traffic Flow Forecasting Using a Deep Convolutional Neural Network," *Sensors*, vol. 20, no. 2, 421, 2020, doi: 10.3390/s20020421.
13. Ultralytics. (2024). YOLO models documentation. Retrieved from https://docs.ultralytics.com/models/yolo/
14. COCO Dataset. (2024). Common Objects in Context. Retrieved from https://cocodataset.org